A Report to the Federal Trade Commission on Reponses to their Request For Information on Establishing a National Do Not E-mail Registry

May 10, 2004

Dr. Aviel D. Rubin
Professor, Computer Science
Johns Hopkins University

Introduction

The Controlling the Assault of Non-Solicited Pornography and Marketing Act of 2003, Pub. L. No. 108-187 (Dec. 16, 2003) (the "CAN-SPAM Act") required that the Federal Trade Commission create a plan and timetable for establishing a National Do Not E-mail Registry. In compliance with this requirement, the Federal Trade Commission put out a request for information (RFI) soliciting proposals for setting up a Do Not Email registry for consumers to register that they do not want to receive unsolicited commercial email. Marketers who wish to send such email would be required to register with the service, and the system would attempt to ensure that registered consumer email addresses would not receive the unsolicited commercial email from registered marketers. This report is an analysis of the submitted proposals. As many of the proposals are confidential and proprietary, and as this report is to be made public, the details of the various schemes as well as the identities of the proposers will not be discussed. Rather, the report focuses on the feasibility of the proposed do not email registry in terms of performance, functionality, privacy, and security.

Designs

There are two basic designs for the do not email registry in the RFI responses. All of the proposed systems have some elements in common. Users register with the Do Not Email registry via a Secure Socket Layer (SSL) protected web page. This is what is typically called a *secure* web page, usually represented by a lock somewhere on the browser window, along with other visual feedback that the web page is protected. On the web site, users enter their email address and their preferences, such as indicating that they do not want to receive unsolicited commercial email. The registry then sends an email to that address requesting confirmation that the user has registered. At that time, credentials, such as a password and/or PIN can be established to enable the user to view their status in the registry and to change their settings. This process is consistent with the way e-commerce sites typically operate today and represents a well-known and usually accepted level of risk for consumers. Considering that there is no previous relationship between consumers who are registering and the registry, this is perhaps the best way to register a large volume of users.

In all of the proposed designs, marketers register via a similar process, but they are required to prove more about their identity. Again, there are standard procedures in the industry for registering marketers in a system such as this over an SSL web connection.

Third Party Forwarding

One proposed design requires all unsolicited commercial email to pass through the registry service. The service itself compares the destination email addresses to the addresses in the database of registered consumers. It then drops messages intended for registered users and allows through messages for other addresses that did not register to opt out. We will call this approach the remailing approach. In all cases marketers' messages are marked so that end users who did not register could still filter the messages. This would allow users to maintain at least two different mailboxes, one for registered email and one for completely unexpected email. The idea is that the first mailbox would only contain desirable email and that the second mailbox would contain email from unknown people as well as noncompliant spam.

The point of the remailer approach is that it does not reveal information to the marketers about which addresses on their lists are "live" addresses. Such information is valuable to spammers. If the registry had to be designed to scale to the size of the email address database, then it might be feasible. Assuming the database contained 300 million addresses, then it could be broken up into a distributed system where different nodes handled different portions of the list. The task is parallelizable, so this would not represent a real problem. So, under the false assumption that the system scales to the number of entries in the registry, with enough hardware and bandwidth the remailing service could be a reality. But, there are problems with this logic. Malicious parties intent on disruption could populate the database with many bogus email addresses, increasing the cost of maintaining it.

However, the scheme proposed in the RFI response requires all email to go through the remailer, even non-marketing messages, such as personal email between two people. This is a requirement of the scheme to prevent non-compliant spammers from bypassing the system. Thus, the system must scale to the total amount of email sent on the Internet, not just to the size of the registry. In this case, the third party forwarding service would represent a choke point for email and a single point of failure for the most popular application on the Internet. It is totally unreasonable to expect that all of the email in the US could be handled by a single provider. It is exactly counter to the distributed nature of the Internet which has a natural load balancing property. Today, anyone can set up a mail server, and anyone can act as a mail relay. As the use of the Internet has grown, so has its capacity to handle the increased volume of email because new entities that came online provided their own mail services.

The Internet is robust to many forms of denial of service because when nodes go down, traffic gets routed around them. If all email were required to go through a single entity, regardless of that entity's size, we would lose all of the advantages of the current Internet architecture. A single denial of service attack against that entity would completely disable email for all people in the US. This is a tremendous risk. Furthermore, the entity running the forwarding service would represent a huge target for attackers and is likely to be hit hard and often, and service disruption is likely to happen with some frequency. In fact, we have come to rely on email for so many things, that it could be considered part of our critical infrastructure and a remailer that handled all email traffic would represent a target for terrorists, as well as traditional hackers.

A third party forwarding service that handled all email would have access to all of the messages sent between parties on the Internet. That represents a tremendous ability for corporate espionage and for privacy compromise. Since most email on the Internet today is not encrypted, the party providing this service would be able to read all email messages. Sophisticated search scripts and data mining techniques could enable tracking all the messages of a particular user, or all of the email on a particular topic.

It is unrealistic to assume that all email will be routed through the third party. There are many independent and autonomous entities on the Internet who run standard mail protocols, and whose mail service will function, even if there is a requirement to go through the third party. Anyone who wants to can simply ignore the service. Without a fundamental overhaul of the Internet architecture and the Internet protocols, there is no way to mandate that all email go through the remailing service. Such fundamental architectural and protocol level changes are not going to happen.

There is another risk to the remailer approach. In this proposed solution, the third party running the service has in its possession an enormous list of email addresses that were live at some point. It is likely that if people bothered to register their addresses, that the number of live addresses actually in use by people is high. (If the number of live addresses is low, then the registry is not providing any real benefit anyway.) If the list contained hundreds of millions of addresses, it would be worth millions of dollars. Such a valuable list is likely to leak out, even in the face of the most stringent security measures. No matter what safeguards are in place, the list would be accessible to several people because it would need to be used in real time as marked email messages are processed. The technicians with access to the list are not likely to have millions of dollars, and if past history is any indication, the most likely scenario is that a corrupt insider would compromise the list. If it is ever compromised, there is no recovering from that. There is no way to undo the exposure of the list, and if spammers ever got their hands on it, the email addresses would clearly be inundated with spam.

The RFI responses overstate the value of keeping the email addresses in the registry in hashed format. Hashing is described in detail in a section below. While a hashed list would be worth less than an unhashed list, there is always a risk that an operator could capture the email addresses before they are hashed. Insider threat is a serious problem in any system, and in the case of a Do Not Email registry, a malicious or disgruntled insider could easily capture the plaintext list of addresses. If this service were running for some time, it is more likely than not that the plaintext addresses would leak at some point, given the history of computer security incidents.

Besides the all of these risks, the remailer solution would not be effective at reducing spam. This is because noncompliant spammers will disguise spam messages as personal messages and will not tag them as unsolicited commercial email. Such tactics are common practice these days. The remailer approach thus introduces serious threats to email availability while not offering any practical benefit.

Scrubbing

Another approach to establishing a Do Not Email registry is the *scrubbing* approach. In this approach, a third party maintains a list of all the registered email addresses. Marketers submit their list of email addresses to the registry, and the registry returns the addresses from the original list that were not registered as wanting to avoid unsolicited commercial email. A negative side effect of this approach is that it enables registered marketers to build a list of email addresses that belong to people with live addresses. All of the addresses that were removed from the submitted list belong to people who have registered. While the marketers might not risk sending spam to those addresses, it is certain that the less reputable ones sell the list of live addresses to less scrupulous spammers or ones who reside outside of the jurisdiction of US law.

Just as in the remailing approach, the scrubbing approach requires that a list of live email addresses exist. While the party owning that list may be well intentioned, it is unlikely that such a valuable list would not leak out. History is replete with insider attacks, as well as external break-ins to highly sensitive sites, such as the Pentagon computers. The Do Not Email Registry represents the kind of *prize* that attracts hackers. In this case, the prize has monetary value as well. Once the list is exposed, there is no way to undo it. Potentially millions of people would have to change their email addresses or be subjected to enormous amounts of spam that dwarf the amount received today, since today spammers do not have access to such a huge list of live addresses.

The limitations of proposed safeguards

The RFI responses propose certain safeguards to protect the Do Not Email registry. Unfortunately, none of these is actually capable of providing the desired protection, as described in this section.

Hashing the list

One idea put forth in the RFI responses (and mentioned in the RFI itself) is that the e-mail addresses could be cryptographically hashed so that if the database were compromised, the attacker would not obtain the list of addresses. The idea is that when the list is used, candidate email addresses are also hashed and compared to the stored values. If they match, the address is in the list, and if they do not, then the address is not in the list. Furthermore, the suggestion is made to prepend a random *salt* value so that without the salt, attackers who obtain the list will not be able to even check to see if a candidate email address is on the list.

It should also be pointed out that hashing provides absolutely no security against a marketer who obtains a scrubbed list and uses that to sell the addresses that were scrubbed by the registry. Whether or not the list is hashed has no impact on a malicious marketer in the scrubbing approach. The point of hashing is to attempt to avoid compromise of the email addresses if a hacker or an insider obtains the data in the registry in its stored format.

Without hashing, a compromise of the registry database results in exposure of all of the registered email addresses. This is a total disaster. However, even exposure of a hashed list is a catastrophe. A spammer with a copy of a hashed list of email addresses is able to find out, for any email address, if the address is in the registry. The attacker simply hashes a candidate email address and sees if the hashed value is in the list. This is very powerful. Furthermore, hashing is a very efficient operation. On a standard 1 Ghz Pentium computer, an attacker could hash millions of email addresses in seconds. So, regardless of the size of the compromised registry, and regardless of the number of candidate email addresses, an attacker could easily determine which ones are live in a short amount of time, with no overhead or cost. No real sophistication would be needed. Anyone who knows how to do simple script programming could develop the programs needed to launch this attack in a matter of several hours or less.

Besides guessing from a candidate list, an attacker is able to generate candidate addresses, using common names in well-known domains to try to produce email addresses and then to hash them and compare them to the registry. Such techniques, called dictionary attacks, are well developed, and there is publicly available software to aid in this task. Here's how a dictionary attack works. The attacker obtains a *dictionary* of known terms in a particular area of knowledge. In this case, the list contains first and last names. To generate email addresses, the program might consider the domain names of all universities. For example, the University of Michigan is `umich.edu`, and Johns Hopkins Univesity is `jhu.edu`. The program would then generate a huge list of every possible name at those domains. The program could start with last names, then first names followed by a dot, and followed by the last name, and then the first letter of the first name, followed by the last name. These are very common ways of creating email addresses. For each name, the attacker would append `@umich.edu` or `@jhu.edu`, and then additional schools could be added. This could be attempted for `aol.com`, `msn.com, yahoo.com` and other popular ISPs. Here is an example of how the list might look:

```
aalto@umich.edu
aaltonen@umich.edu
ahlberg@umich.edu
aho@umich.edu
…
zonder@umich.edu
zoro@umich.edu
aaron.aalto@umich.edu
…
zach.zoro@umich.edu
```

```
aaalto@umich.edu
aaaltonen@umich.edu
aaho@umich.edu
…
zzoro@umich.edu
aalto@jhu.edu
aaltonen@jhu.edu
…
zzoro@jhu.edu
```

A program running on a 1Ghz Pentium computer could generate hundreds of millions of such addresses, hash them, and compare them to the list in minutes. The program could be left to run indefinitely, recording every time it generates a live email address.

In the Computer Security field, it is well known that insider attacks account for the most loss in terms of proprietary data. While we have well-developed techniques for thwarting external attackers, for example, firewalls, intrusion detection systems, and virtual private networks, the state of the art at protecting against malicious insiders is currently dismal. Proprietary algorithms, code, and designs leak all the time. Industrial espionage is rampant, and theft of data by people with legitimate access is the most common form of loss known to today's corporations. This is why the hashed list of email addresses, which is such a valuable target, is almost certain to be compromised at some point if a Do Not Email registry is deployed. The technology does not exist to protect it against insiders. Furthermore, spot checks of the external security of many sites, even highly sensitive ones, often reveal huge vulnerabilities. This is why many corporate and military networks get invaded by external hackers all the time.

Salting

A common technique used when hashing data to avoid its exposure in the case of compromise is called *salting*. The idea is to prepend the data with a random string (typically 128 bits long) and then to hash. The salt cannot be kept secret, as it is needed every time a comparison is made. This technique is most useful when storing lists of passwords so that duplicate entries in a list will not appear the same. Each password has a unique salt that is stored with the hashed password. Thus, if two users pick the same password, the salted, hashed password will still be different. The technique does not really apply to the email address storage problem. The salt must be prepended to the candidate email addresses before they are hashed and compared to the list. Anyone, such as an insider or hacker, who obtains the stored, hashed list of email addresses is also in a position to obtain the salt. So, salting does not add any additional security for the Do Not Email registry.

Canaries

Canaries, sometimes called *decoys* or *honeytokens*, are data elements stored in a database that have no legitimate use. Adding such information is a common technique used to identify data leaks. If there is a value in a database that cannot possibly be a legal value, and that data element is found somewhere outside of the system, then there is good reason to believe that the secrecy of the database has been compromised. This technique is not very effective in the domain of the Do Not Email registry. It will do nothing to improve the security of the database, nor to prevent spam of compromised email addresses if the database leaks.

To apply this technique to the registry, invalid random looking email addresses (canary addresses) such as `hb43252dkwoie@blebedoci345.com` could be added to the database. The address should be chosen so that it is not something that would be produced by a dictionary attack, as described above. The registry owner would purchase the domain `blebedoci345.com` and monitor it for email messages. If the database were hashed, then no dictionary attack would ever discover the canary address. If email were ever received at that address, the only possible conclusion would be that the prehashed database had been compromised, and that would indicate an insider attack. For a hashed registry, then, canaries can only be used to detect a sophisticated insider attack that would devastate any system. Thus, canaries are useless when dealing with a hashed registry.

If the database is not hashed and email is received at a canary address, then clearly the database is compromised through internal or external attack. The reason is that no scrubbing marketer could have learned of this address when comparing a list of candidate addresses to the registry. Knowing that the registry has been compromised is not very valuable information. While spam received on a canary address indicates that there has been a compromise of the registry, at that point, it is too late to do anything about it. In all likelihood, all of the addresses have already been compromised, and the owners of those addresses will realize this when they start to get flooded with spam. So, canaries are not useful in the context of a Do Not Email registry.

Phishing

The web-based registration process is vulnerable to *phishing* attacks. Phishing is a term used to describe the situation where an attacker sets up a bogus web site that looks and feels like a particular site and draws web traffic there. Unsuspecting users interact with the site the same way that they would with the legitimate site, but in fact, their traffic goes to an attacker. The attacker could thus collect live email addresses. Phishing could be successful through attacks against the Domain Name Service, or simply by registering a domain that sounds legitimate. A variant of this attack is for the attacker to set up a connection to the actual registry web site and act as a man in the middle, registering the user. Although the user would be registered (and the response to the email confirmation would work), the attacker would still see the email address. This would prevent the registry from noticing a drop-off in registrations while the attack was in progress. Carefully logging and monitoring could discover the fact that registrations were coming from a single point. However, if the attacker were distributed and careful, it could become very difficult to notice anything. In addition, as with privacy compromises, there is no way to recover from the fact that an email has been compromised. It cannot be un-compromised.

Tracing non-compliant spammers

A reality that makes the registry ineffective is that most spam comes from parties who would not be compliant. In fact, the vast majority of spam originates from hosts who are either untraceable, traceable to unwitting users who do not realize the spam is coming from their machines, or outside of the US. None of the RFI responses offer any convincing plan for tracing non-compliant spammers. This is due to the fact that there are too many ways that unsolicited e-mail can be sent without traceability.

One way that email can be sent without traceability is through open relays on the Internet. Open SMTP relays do not require any authentication and simply forward email from anyone to any destination. It is possible to disguise email as originating from the relay and not from the actual originator. There has been an effort by the Internet community to limit the number of open relays, but they continue to exist in abundance. Besides open relays, open proxies can be used to forward email and disguise its origin. Open proxies are general purpose forwarding agents. Typically, proxies are designed to help with the security of a site by intercepting all traffic and processing it before letting it through a firewall. However, proxy misconfiguration is common and results in general purpose forwarding that is utilized by hackers and spammers. It is well known that these malicious parties share information with each other on where open proxies exist and how to exploit them.

Worse than the open relays and open proxies that are found on the Internet are SMTP relays that are deposited intentionally by hackers or viruses. These are often referred to as zombies or zombie drones, and are a form of Trojan Horse. A Trojan Horse is a malicious program that is installed on an unsuspecting user's machine with the express purpose of performing a malicious action at some future time. Such programs have become extremely common, especially on Windows machines. The zombies are used for two primary purposes, to launch denial of service attacks, and to relay spam. Spam that is relayed by zombie drones leaves no trace beyond the zombie. Thus, the spam can only be traced back to the machines infected with the zombie programs. The owners of these machines are not aware that they are contributing to spam. The spread of zombies via viruses is totally out of control on the Internet, and it is only getting worse. Any Windows machine that is connected to the Internet and used for e-mail and Web browsing can be expected to be infected with zombies within days. If the machine is used for file sharing, such as with Kazaa or other file sharing programs, the number of zombies on expected to be found on such a machine goes up dramatically. One of the most dramatic trends on the Internet is the proliferation of hacker tools and programs specifically designed to aid in spam. After all, it is one of the few avenues for making real money out of hacking. SMTP relays under the control of hackers can be found all over the Internet.

When spam is relayed from a zombie running an SMTP server, the server removes all previous `Received:` headers, and so the spam that is received gives no hints as to the actual origin. Once spam is received this way, the best course of action is to trace it back to the machine with the zombie, and to try to get the zombie removed. However, it is likely that if a zombie was deposited on a user's computer, that if the zombie were removed, it would come back quickly. The only way to prevent it would be to eliminate the root cause of the vulnerability, which would mean figuring out how the zombie arrived in the first place. This level of forensics and recovery is time consuming and resource intensive, and given the level of infection on the Internet, it is impractical to attempt to eliminate zombies to an extent that would reduce SMTP relaying.

A large amount of spam comes from overseas. In addition, many unsolicited commercial emails come from unwitting users whose machines are under the control of a virus or some other malicious code. Some spammers are utilizing widespread free hot spots, where they can hide behind a WiFi network and launch spam that is untraceable to them. In addition, it is common, especially in large cities, to find open wireless networks from people's apartments or offices. These network provide a perfect anonymous cover for someone to send out thousands of unregistered spam messages without detection.

Conclusions

A Do Not Email registry will do little to slow the flow of unwanted messages from unscrupulous fly-by-night vendors of such things as online medications, pornography, and get rich quick scams. In fact, it is possible that the existence of a Do Not Email registry could actually increase the amount of spam on the Internet. Phishing attacks could fool users into revealing their email addresses to malicious parties, and a compromise of the sensitive database is disastrous.

After examining the responses to the RFI, it is the opinion of this author that the risks outweigh the potential benefit of establishing a Do Not Email registry, and it is therefore recommended that the Federal Trade Commission not pursue this idea.

Author Biography

Aviel D. Rubin is Professor of Computer Science and Technical Director of the Information Security Institute at Johns Hopkins University. Prior to joining Johns Hopkins Rubin was a research scientist at AT&T Labs. Rubin is author of several books including Firewalls and Internet Security, second edition (with Bill Cheswick and Steve Bellovin, Addison Wesley, 2003), White-Hat Security Arsenal (Addison Wesley, 2001), and Web Security Sourcebook (with Dan Geer and Marcus Ranum, John Wiley & Sons, 1997). He is Associate Editor of ACM Transactions on Internet Technology, Associate Editor of IEEE Security & Privacy, and an Advisory Board member of Springer's Information Security and Cryptography Book Series. Rubin serves on the board of directors of the USENIX Association and on the DARPA Information Science and Technology Study Group. He is co-author of a report showing security flaws in a widely used electronic voting system that focused a national spotlight on the issue. Rubin also co-authored an analysis of the governments planned SERVE system for Internet voting for military and overseas civilians, which led to the cancellation of that dangerous project. He is a recipient of the 2004 EFF Pioneer Award. His home page is http://avirubin.com/.